Arduino verstehen und anwenden

Teil 19: Messtechnik – von der Spannungsmessung zum Logic-Analyzer

Bereits in Teil 6 dieser Artikelserie („Sensortechnik und Messwerterfassung" im ELVjournal Okt/Nov 2014) wurden einige Grundlagen der Erfassung von messtechnischen Größen diskutiert.

In diesem Beitrag sollen nun auch komplexere Messaufgaben betrachtet werden. Zunächst stehen einfache Strom-, Spannungs- und Widerstandsmessungen im Vordergrund.

Im Anschluss sollen

- ein Kapazitätsmessgerät und
- ein Halbleitertester

sowie Geräte mit zeitaufgelöster Messwerterfassung wie etwa

- ein Oszilloskop oder auch
- ein Logic-Analyzer

diskutiert werden.

Messung von Spannungen, Strömen und Widerständen mit dem Arduino

Die prinzipiell einfachste Messung ist hier die Spannungsmessung. Mit dem Analog-digital-Wandler eines Arduinos können Spannungen zwischen 0 und +5 V direkt gemessen werden. Der Arduino verfügt hierfür über eine interne Spannungsreferenz. Für diese können mehrere Optionen gewählt werden:

- DEFAULT: voreingestellte Analog-Referenz von 5 V (auf 5 V Arduino-Boards) oder 3,3 V (auf 3,3 V Arduinos)
- INTERNAL: interne Referenz:
  - 1,1 V für ATmega168 oder ATmega328 (z. B. Arduino Uno)
  - 2,56 V für den ATmega8 (ältere Arduino-Varianten)
  - INTERNAL1V1: 1,1 V Referenz (nur Arduino Mega)
- INTERNAL2V56: eingebaute 2,56-V-Referenz (nur Arduino Mega)
- EXTERNAL: für den Anschluss einer externen Referenzspannung am AREF-Pin (0–5 V)

Hinweis:
An den AREF-Eingang dürfen nur Spannungen zwischen 0 und 5 V angelegt werden! Mehr als 5 V oder negative Spannungen können den Arduino zerstören.

Soll eine externe Referenz zum Einsatz kommen, muss diese im Sketch ausgewählt werden, bevor analogRead() aufgerufen wird, sonst werden interne und externe Referenz kurzgeschlossen, was zu einer Zerstörung des Arduino-internen Controllers führen kann.

Alternativ kann man AREF über einen Schutzwiderstand anschließen, dann ist allerdings zu beachten, dass die externe Referenz mit ca. 32 kΩ belastet wird.

Benutzt man die interne Referenz, muss man berücksichtigen, dass diese bei einem Nominalwert von 1,1 V ± 10 % zwischen 1,0 und 1,2 V liegt. Für präzise Messungen ist es daher erforderlich, den genauen Wert mit einem Referenzvoltmeter zu bestimmen.

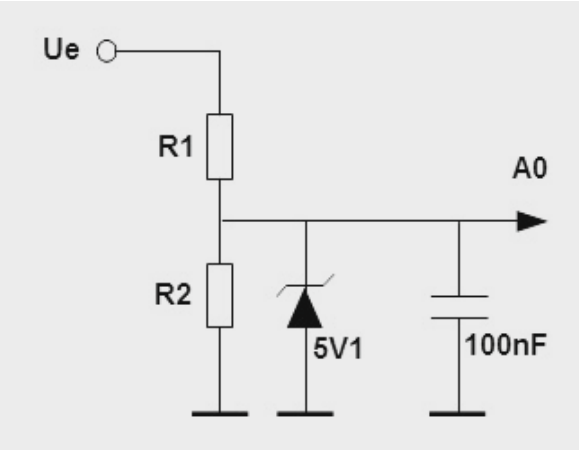


Bild 1: Eingangsschaltung für das Arduino-Voltmeter

Möchte man Spannungen bis zu beispielsweise 20 V messen, ist ein Spannungsteiler erforderlich. Bild 1 zeigt einen entsprechenden Aufbau. Aufgrund ihrer höheren Präzision sollten hier Metallschicht-Widerstände (± 1 % Toleranz) eingesetzt werden. Zusätzlich wurde noch ein 100-nF-Kondensator eingefügt, um stabile Messwerte zu erhalten. Eine 5V1-Zenerdiode schützt den Arduino-Eingang vor unerwünschten Überspannungen.

Der folgende Aufbau erlaubt mit

R1 = 1 MΩ
R2 = 50 kΩ

die präzise Messung von Spannungen zwischen 0 und 20 V. Der Wert von 50 kΩ entsteht durch Parallelschaltung von zwei 100-kΩ-Widerständen.

Für präzise Ergebnisse müssen die Widerstandswerte genau vermessen und die Ergebnisse in den Sketch eingesetzt werden. Gleiches gilt für die interne Referenzspannung.

Eine Mittelung über 10 Werte führt zu einer weiteren Verbesserung der Präzision, da sich statistische Messfehler der Einzelmessungen herausmitteln. Der vollständige Sketch sieht damit wie folgt aus:

```
// digital voltmeter

const int analogInPin = A0;
const float U_ref = 1.035;  // in V
const float R2 = 49.8;      // in kOhm
const float R1 = 996;       // in kOhm
const int averages = 10;

float voltage;
int ADC_count;

void setup()
{ Serial.begin(9600);
  analogReference(INTERNAL);
}

void loop()
{ ADC_count = 0;
  for (int i = 0; i<averages; i++)
  ADC_count += analogRead(analogInPin);

  Voltage =
  (R1+R2)/R2*U_ref*ADC_count/1023/averages;

  Serial.print("\t voltage = ");
  Serial.println(voltage, 1);

  delay(300);
}
```

Mittels eines geeigneten Shunts können so auch Stromstärken bestimmt werden. Widerstandsmessungen sind ebenfalls möglich. Hierzu muss lediglich einer der Widerstände im Spannungsteiler als Referenz, der andere als Messobjekt definiert werden. Die Eingangsspannung muss in diesem Fall natürlich ebenfalls bekannt sein. Man kann hier auch auf die 3,3 V des Arduinos zurückgreifen.

Bestimmung von Kapazitäten und Halbleiterparametern

Bedingt durch ihren inneren Aufbau weisen Elektrolytkondensatoren eine vergleichsweise hohe Ausfallwahrscheinlichkeit auf und zählen deshalb zu den problematischsten Bauelementen der Elektronik.

Meist liegt die Hauptursache für den Ausfall im Austrocknen des Elektrolyts und dem damit verbundenen erheblichen Kapazitätsverlust. Viele Ausfälle von Netzteilen, PCs oder Audioverstärkern sind auf dieses Problem zurückzuführen.

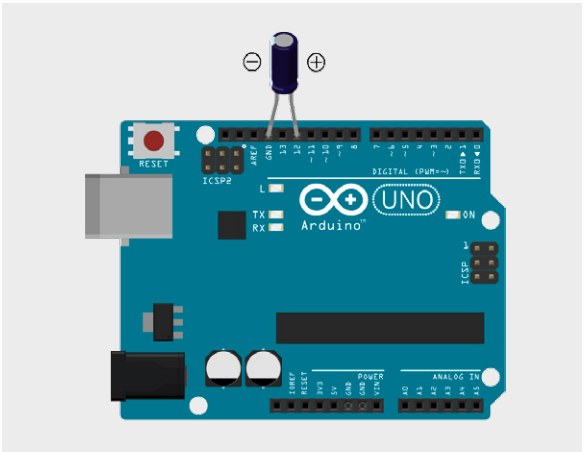


Bild 2: Arduino als Kapazitätsmessgerät

Mit dem Arduino kann die Kapazität eines Elkos ohne zusätzliche Hardware mit guter Genauigkeit bestimmt werden (siehe Bild 2).

Achtung:
Kondensatoren sind vor dem Anschluss an den Arduino unbedingt zu entladen, da die Digitaleingänge sonst zerstört werden könnten!

Das Messprinzip ist simpel: Zunächst wird der Pluspol des Elkos auf LOW-Potenzial gelegt. Danach wird dieser Pin als Eingang geschaltet, d. h. der Pin wird hochohmig. Wenn nun dieser Input-Pin den Befehl

digitalWrite(activePin,HIGH);

erhält, wird dadurch der interne Pull-up-Widerstand des Arduinos aktiviert und der Kondensator lädt sich über diesen Widerstand auf. Dabei wird gemessen, wie lange es dauert, bis sich der Kondensator auf etwa die halbe Versorgungsspannung aufgeladen hat. Diese Zeit ist ein direktes Maß für die Kapazität des Kondensators:

$$T = R * Cx$$

R ist hier der interne Pull-up. Dieser hat einen Wert von ca. 60 kΩ. Durch die Kalibrationskonstante „cal" wird die Ladezeit direkt in einen Kapazitätswert in nF umgerechnet. Ist der berechnete Wert größer als 1000, erfolgt die Anzeige automatisch in µF. So lassen sich nicht nur Elkos im µF-Bereich vermessen, sondern auch kleinere ungepolte Kondensatoren bis zu einem Minimalwert von ca. 10 nF. Bei noch kleineren Kapazitäten werden die Messzeiten zu kurz und das Ergebnis ungenau.

Im Programm wird zunächst die Kalibrationskonstante „cal" definiert. Durch Vermessung genau bekannter Kapazitäten kann diese Konstante nachjustiert werden. In der Main-Loop wird die bereits oben beschriebene Messprozedur ausgeführt. Schließlich wird der ermittelte Messwert unter Verwendung einer Autorange-Funktion auf die serielle Schnittstelle ausgegeben.

```
// Capacity meter

float cal = 0.5;              // calibration constant
int activePin = 12;
float chargingTime = 0.0;
float capacity = 0.0;

void setup()
{ Serial.begin(9600);}

void loop()
{ pinMode(activePin,OUTPUT);
  digitalWrite(activePin,LOW);
  chargingTime=0.0;
  delay(1000);

  pinMode(activePin,INPUT);
  digitalWrite(activePin,HIGH);

  while(!digitalRead(activePin)) chargingTime++;
  capacity=chargingTime*cal;

  if(capacity < 1000)
  {   Serial.print(capacity); Serial.println(" nF");
  }

  else
  { capacity/=1000;
      Serial.print(capacity); Serial.println(" uF");
  }
  delay(1000);
}
```

Aber nicht nur passive Bauelemente können mit einfachen Mitteln ausgemessen werden. Auch aktive Komponenten wie Transistoren können mit dem Arduino geprüft werden. Die Schaltung nach Bild 3 misst die wichtigsten Transistorparameter.

Die Basisspannung Ube liefert eine Aussage über das Transistorgrundmaterial:

Ube ≦ 0,3 V: Germaniumtransistor
Ube ≦ 0,7 V: Siliziumtransistor

Aus der Kollektorspannung Uce lässt sich der Kollektorstrom Ic berechnen, entsprechend aus Ube der Basisstrom Ib. Damit ergibt sich nach

B = Ic / Ib

die Gleichstromverstärkung B des Transistors. Typische Werte für Kleinsignaltransistoren liegen im Bereich von 100 bis ca. 800.

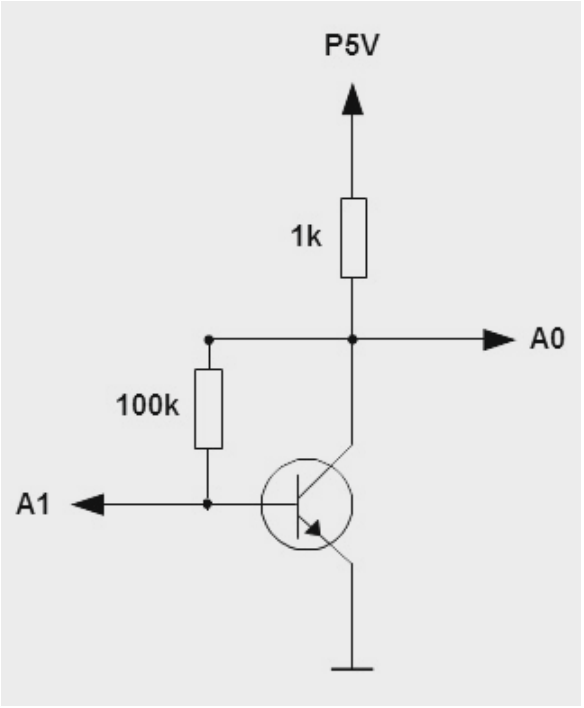


Bild 3: Transistortester

```
// Transisortester

float Uce, Ube, beta;

void setup()
{  Serial.begin(9600);
}

void loop()
{ Uce=5.0*analogRead(0)/1023;
  Ube=5.0*analogRead(1)/1023;
  beta = (5-Uce)/(Uce-Ube)*100;

  Serial.print("Ube = ");  Serial.println(Ube);
  Serial.print("Ib = ");  Serial.print((Uce-Ube)*10); Serial.println(" uA");
  Serial.print("Ic = ");  Serial.print(5-Uce); Serial.println(" mA");
  Serial.print("beta = "); Serial.print(beta); Serial.println(" @");
  Serial.println();

  delay(1000);
}
```

Das Programm ermittelt die Werte für Uce und Ube. Daraus werden die Parameter für Kollektor- und Basisstrom berechnet. Abschließend werden die Ergebnisse auf die serielle Schnittstelle ausgegeben.

Die Anwendung ist sehr praktisch, wenn man eine größere Anzahl von Transistoren prüfen will. Man erhält hier sehr schnell nicht nur die Information, ob ein spezieller Transistor funktionsfähig ist, sondern auch den genauen Wert der Stromverstärkung. Es ist also mit diesem Sketch leicht möglich, Transistoren gleichen Typs nach ihrer Stromverstärkung zu sortieren. So kann man etwa für Stromspiegel oder ähnliche Applikationen leicht sogenannte „matched pairs" selektieren, d. h. Transistorpärchen mit möglichst identischen Stromverstärkungen.

Oszilloskop und Logic-Analyzer

Nach dem Multimeter ist das Oszilloskop das wichtigste Messgerät im Elektronik-Labor. Wenn man häufig mit der Analyse digitaler Schaltungen oder von Mikrocontroller-Projekten befasst ist, dann weiß man auch einen Logic-Analyzer zu schätzen. Damit lassen sich problemlos z. B. Codes von IR-Sendern, Signale auf seriellen Schnittstellen, I^2C-Bussen oder die Übertragungsprotokolle von Funksignalen überprüfen.

Sowohl ein Oszilloskop als auch ein Logic-Analyzer lassen sich mit einem Arduino aufbauen. Natürlich können diese Arduino-basierten Geräte nicht mit kommerziellen Gegenspielern mit Anschaffungskosten von € 10.000,– und mehr mithalten. Dennoch ist es erstaunlich, was man auch mit sehr einfachen Mitteln erreichen kann. Für den Hobbybereich, aber auch für einfachere professionelle Einsätze sind die erreichbaren Leistungsmerkmale dieser Arduino-basierten Messgeräte häufig durchaus ausreichend.

Für den Arduino Uno existieren verschiedene Oszilloskop-Emulatoren wie etwa Girino oder Arduino-Scope. Eine sehr gute und leistungsfähige Anwendung ist XOscillo. Dieses Programmpaket kann unter

http://code.google.com/p/xoscillo/

kostenlos aus dem Internet geladen werden. Damit werden die folgenden Leistungsmerkmale erreicht:

- Es ist keine zusätzliche Hardware erforderlich
- Maximale Abtastfrequenz: 7 kHz
- Bis zu 4 Kanäle (bei entsprechend niedrigerer Sample-Rate)
- 8 bit vertikale Auflösung
- Variable Trigger-Spannung auf Kanal 0
- Theoretisch unbeschränkte Anzahl von Messpunkten

Sicher liegt die maximale Abtastfrequenz von 7 kHz im Vergleich zu einem kommerziellen Speicheroszilloskop sehr niedrig. Für einfache Messungen im Niederfrequenz- und Audiobereich ist die XOscillo-Applikation aber durchaus nützlich.

Das XOscillo-Programmpaket besteht aus einem Arduino-Sketch und einer grafischen Benutzeroberfläche für den PC. Nach dem Laden des Sketches auf den Arduino kann die Windows-Applikation (XOscillo.exe) gestartet werden. Danach wird im File-Menü die Option „New Analog Arduino" ausgewählt.

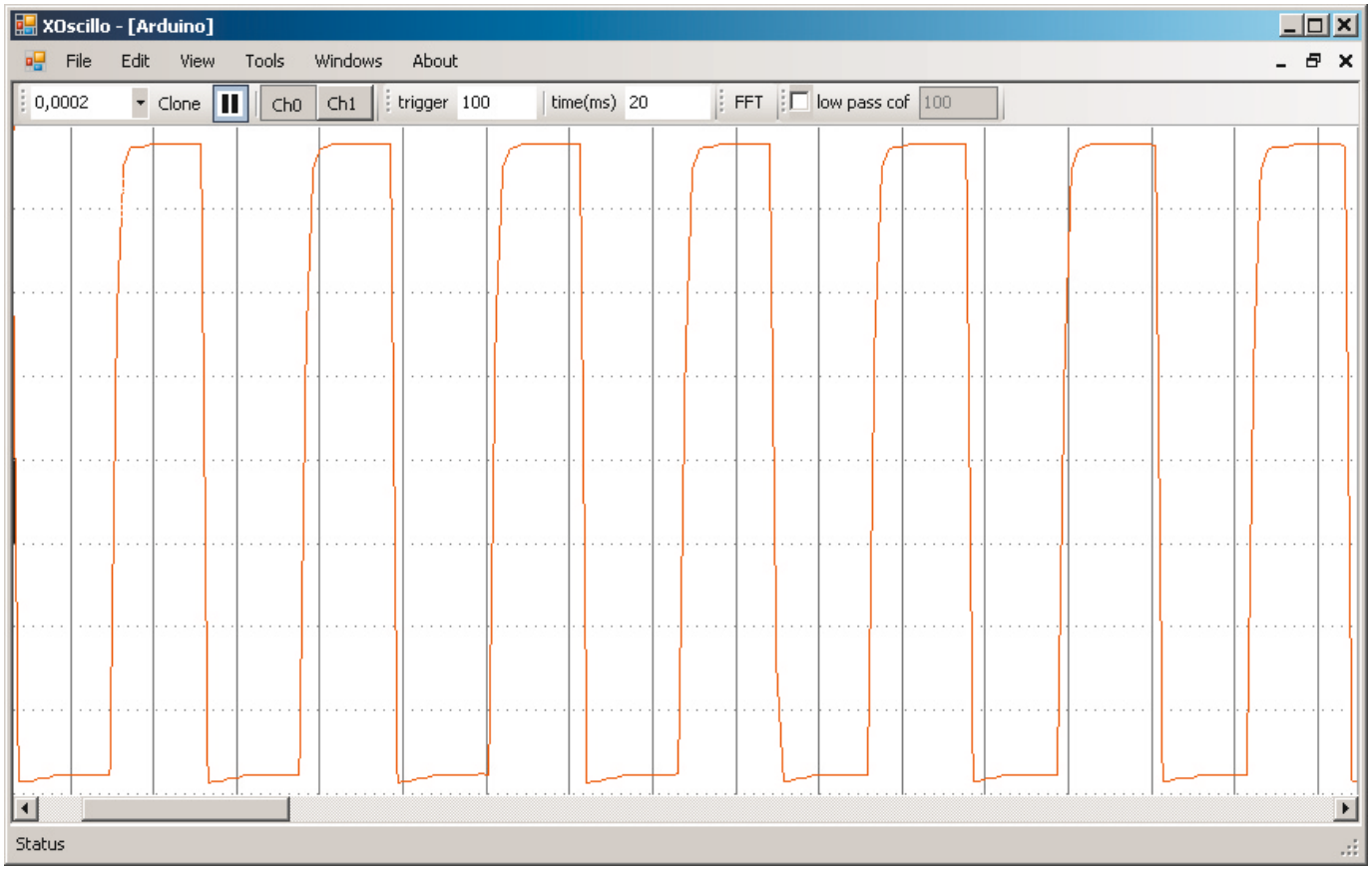


Bild 4: Das Arduino-basierte Oszilloskop zeigt ein Rechtecksignal

Es erscheint die in Bild 4 dargestellte Oberfläche eines **D**igitalen **S**peicher**o**szilloskops (DSO). Mit CH0 (rot) und CH1 (blau) können bis zu zwei analoge Kanäle ausgewählt werden.

Die Zeitbasis kann zwischen 1 s/div und 500 µs/div eingestellt werden. Die Aufzeichnungszeit ist frei in Millisekunden wählbar.

XOscillo verfügt zwar auch über die Möglichkeit zur Aufzeichnung mehrerer Digitalkanäle, jedoch steht für die Logic-Analyse eine noch etwas professionellere Anwendung zur Verfügung. Diese kann unter

https://github.com/gillham/logic_analyzer

geladen werden. Das Programmpaket enthält wieder Arduino-Sketche und Windows-Programme.

Zusätzlich wird das Programm „Logic Sniffer" benötigt. Dieses ist Java-basiert und kann also sowohl unter Windows als auch unter Linux verwendet werden. Für die Verwendung unter Windows muss natürlich Java installiert sein. Unter

https://www.lxtreme.nl/ols/#download

kann die jeweils neueste Version heruntergeladen werden. Jetzt muss nur noch die Datei

ols.profile-agla.cfg

aus dem Download-Paket in den Ordner „plugins" kopiert werden. Dann sollte nach dem Start des Programms die Benutzeroberfläche des Logic-Analyzers erscheinen.

Der Analyzer basiert auf dem SUMP-Software-Modul, welches verschiedene Hardware-Frontends, unter anderem eben auch den Arduino, unterstützt. Bild 5 zeigt die Oberfläche des Analyzers.

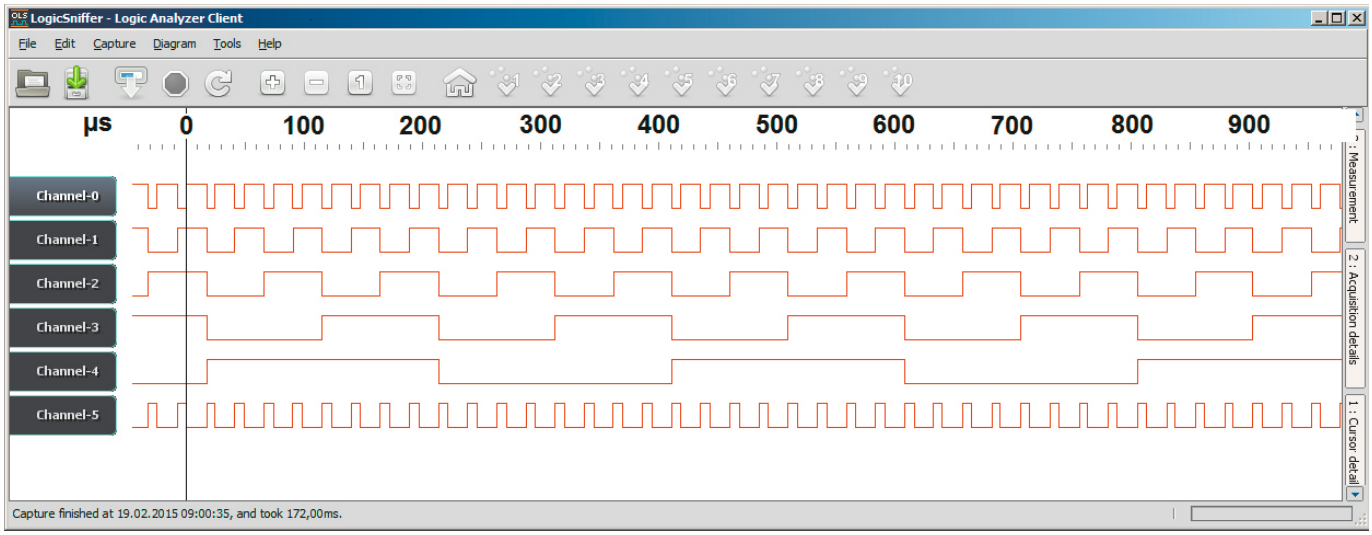
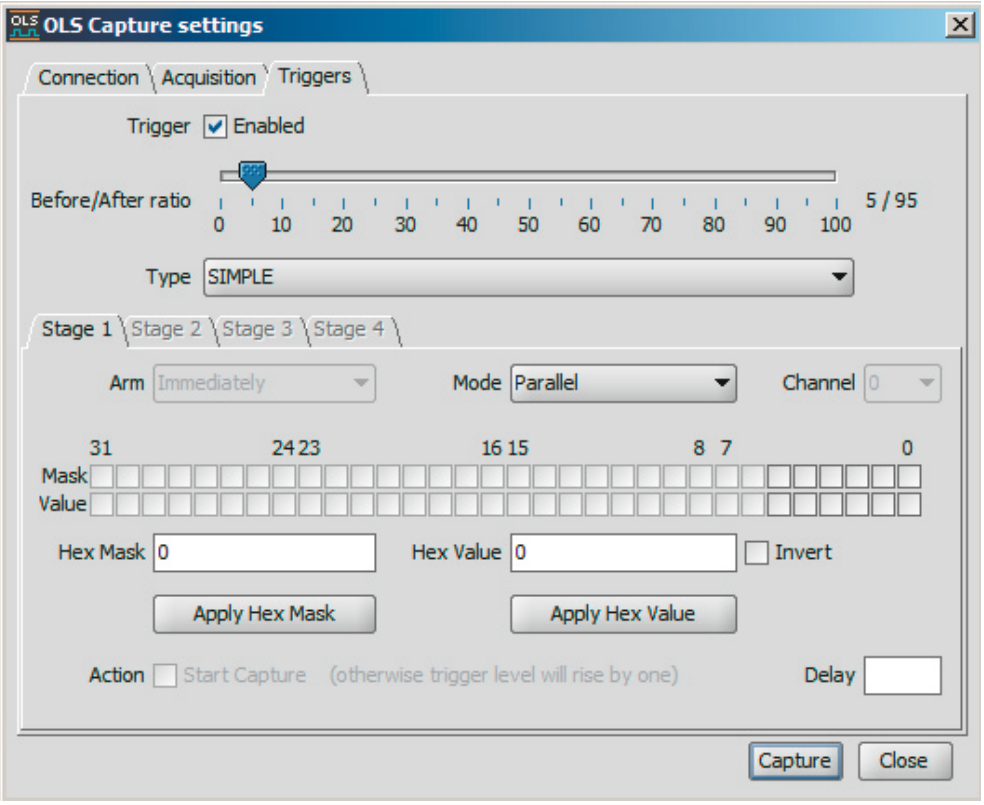


Bild 5: Arduino als 5-Kanal-Logic-Analyzer

Bild 6: Triggermenü Logic-Analyzer



Mit dem Arduino können bis zu 6 Kanäle parallel dargestellt werden. Die Pins D7 bis D13 sind als Eingänge voreingestellt. Natürlich können durch entsprechende Änderungen im Sketch auch andere Pins verwendet werden.

Kanal 0 zeigt hier ein Clock-Signal das jeweils mit einem Flip-Flop in der Frequenz herunter geteilt wird. Die Signale der Kanäle 1 bis 4 weisen daher jeweils die halbe Frequenz auf. In Kanal 5 wird schließlich das invertierte Clock-Signal dargestellt.

Bild 6 zeigt als Beispiel für den umfangreichen Funktionsumfang des SUMP-kompatiblen Analyzers das Triggermenü. Hier lassen sich nicht nur Pre-Triggeroptionen einstellen, sondern auch bestimmte Bitmasken wählen, bei welchen dann die Triggerung ausgelöst werden soll. Ähnlich umfangreich sind auch die Optionen und Möglichkeiten in den anderen Untermenüs. Der Logic-Analyzer lässt sich so sehr vielseitig auch für professionelle Messaufgaben einsetzten.

Ausblick

Die Beispiele in diesem Beitrag zeigen, wie vielseitig Mikrocontroller und damit auch Arduino-Boards einsetzbar sind. Von der einfachen Spannungsmessung bis hin zu Logic-Analyzern mit nahezu professionellen Leistungsmerkmalen können viele Messaufgaben abgedeckt werden.

Nachdem in diesem Beitrag nützliche Anwendungen und Geräte für das Elektronik-Labor im Vordergrund standen, werden sich die nächsten beiden Themen wieder eher um allgemein einsetzbare Technikanwendungen drehen.

Dort soll es nämlich um die Audiotechnik, die Ton- und Klangerzeugung sowie um die digitale Synthesizertechnik gehen. Darüber hinaus wird im Rahmen dieser Beiträge das Abspielen von Audiodateien auch in einem frei programmierbaren, im Eigenbau erstellten MP3-Spieler thematisiert. ELV

Weitere Infos:

[1] Mikrocontroller-Onlinekurs, Franzis-Verlag, exklusiv für ELV, 2011, Best.-Nr. CK-10 20 44

[2] G. Spanner: Arduino – Schaltungsprojekte für Profis, Elektor-Verlag, 2012, Best.-Nr. CK-10 94 45

[3] Grundlagen zur elektronischen Schaltungstechnik finden sich in der E-Book-Reihe „Elektronik!“ (http://www.amazon.de/dp/B000XNCB02)

[4] Lernpaket „AVR-Microcontroller in C programmieren“, Franzis-Verlag, 2012, Best.-Nr. CK-10 68 46

[5] Lernpaket „Physical Computing”, Franzis-Verlag, 2015, Best.-Nr. CK-12 21 81

Preisstellung Oktober 2016 – aktuelle Preise im Web-Shop

Empfohlene Produkte	Best.-Nr.	Preis
Arduino Uno	CK-10 29 70	€ 27,95
Mikrocontroller-Onlinekurs	CK-10 20 44	€ 99,–

Alle Arduino-Produkte wie Mikrocontroller-Platinen, Shields, Fachbücher und Zubehör finden Sie unter: www.arduino.elv.de